

ATLAS
MANUAL DE USUARIO
ARBOL

Versión 1.3

Área de Aplicaciones Especiales y Arquitectura de Software



Agencia de Informática y Comunicaciones de la Comunidad de Madrid



Hoja de Control

Título	Manual de Usuario del componente Árbol		
Documento de Referencia	NORMATIVA ATLAS		
Responsable	Área de Aplicaciones Especiales y Arquitectura de Software		
Versión	1.3	**Fecha Versión**	01/03/2013

Registro de Cambios

Versión	Causa del Cambio	Responsable del Cambio	Fecha del Cambio
1.0	Versión inicial del documento	Área de Integración y Arquitectura de Aplicaciones	12/11/2010
1.1	Se modifica el nombre del Área	Área de Aplicaciones Especiales y Arquitectura de Software	27/09/2011
1.2	Modificación general del documento debido a la creación de un nuevo componente para Atlas 1.2.2.	Área de Aplicaciones Especiales y Arquitectura de Software	24/08/2012
1.3	- Ampliada la información sobre la carga de nodos y estado del árbol. Nuevo atributo *cacheMode*. - Nuevo apartado **4.4 Interacción con el árbol**.	Área de Aplicaciones Especiales y Arquitectura de Software	13/02/2013

Índice

1. INTRODUCCIÓN

Este documento contiene el manual de uso del componente visual ***Árbol*** del framework Atlas. En él se incluye información sobre cómo utilizar dicho componente en una aplicación web, así como información acerca de la configuración de los parámetros fundamentales del componente.

1.1. AUDIENCIA OBJETIVO

Este documento está orientado a toda aquella persona que esté desarrollando una aplicación Web basada en el Framework Atlas y necesite utilizar componentes de presentación en su aplicación Web.

1.2. CONOCIMIENTOS PREVIOS

Para un completo entendimiento del documento, el lector deberá tener conocimientos previos sobre las siguientes tecnologías:

- Java Server Faces (JSF)
- Facelets
- Richfaces
- Spring Framework
- Hibernate

Para saber más sobre dichas tecnologías, consultar el apartado de este documento, *Enlaces Relacionados*.

2. DESCRIPCIÓN

Este componente sirve para mostrar al usuario un conjunto de datos organizados jerárquicamente en forma de árbol. Los nodos de este árbol se pueden desplegar o replegar para mejorar la presentación de los datos al usuario. En caso de desplegar un nodo, su contenido será obtenido mediante una petición tipo AJAX. La implementación de este componente se basa en los componentes de RichFaces: ***rich:tree*** *y* ***rich:treeNode***.

El aspecto visual del componente es el siguiente:



3. INSTALACIÓN Y CONFIGURACIÓN

En este apartado se incluye información sobre la instalación y la configuración del componente árbol.

3.1. INSTALACIÓN

El componente árbol ya viene instalado en el arquetipo Web, incluido con el módulo de componentes visuales. Por este motivo no es necesaria una instalación adicional si se parte del arquetipo.

3.2. CONFIGURACIÓN

Una vez instalado el componente, se debe proceder a su configuración.

La configuración comienza con la definición e inicialización en el backing bean de un objeto de la clase `atlas.componentes.bean.AtlasTree`. Este objeto se encarga de alimentar el árbol y gestionar los diferentes eventos generados y se enlaza con el componente mediante el atributo *dataModel*.

Posteriormente hay que crear los servicios para los diferentes niveles del árbol, estos servicios deben cumplir una interfaz y devolver una lista de objetos que cumplan otra interfaz. A continuación se describen ambas interfaces.

3.2.1. Servicios

Para que el componente obtenga la información de los diferentes niveles del árbol es necesario definir uno o varios servicios. Estos servicios deben estar definidos en el contexto de Spring, y cumplir con la interfaz
atlas.componentes.service.tree.NodesService
Esta interfaz tiene un solo método:
List<? extends Node> getNodes(Object clave).
Cada vez que se expanda un nodo del árbol el componente hará una llamada a este método con la clave correspondiente del nodo pulsado, y se mostrarán aquellos nodos que devuelva el servicio. En el siguiente apartado se definen como deben ser estos nodos.
Se puede especificar un solo servicio para todos los niveles del árbol o uno diferente para cada nivel. En el primer caso el componente no sabrá a priori el número de niveles del árbol, ya que hará llamadas al mismo servicio cada vez que se expanda un nodo, en el segundo caso se entiende que el árbol tiene tantos niveles como número de servicios. Este detalle implica que si se definen varios servicios el componente no hará llamadas una vez llegado al mayor nivel del árbol, sin embargo, si es un solo servicio seguirá haciendo llamadas independientemente del nivel.
El modo de enlazar el componente con el servicio es mediante un atributo en la definición en la página o directamente en el backing bean a través de la clase *AtlasTree.*

Para mostrar el funcionamiento de la configuración del componente *Árbol* se va a mostrar un ejemplo, con tres servicios para dar lugar a un tipo de árbol de tres niveles.

Comenzaremos definiendo el árbol y los niveles que lo forman y por tanto los servicios que necesitaremos:

- Elementos de primer nivel: *las letras del abecedario*. No tiene restricción porque se trata de los elementos de primer nivel, aunque el componente permite especificar una clave inicial, que se utilizará para obtener el primer nivel de árbol del servicio correspondiente.

- Elementos de segundo nivel: *provincias de España*. La restricción con el nivel anterior consiste en que dado un nodo del primer nivel cuya clave consistirá en una letra, entonces, los nodos hijos serán las provincias cuya inicial coincida con el valor de esa letra.

- Elementos de tercer nivel: *municipios de España*. La restricción con el nivel anterior consiste en que dado un nodo del segundo nivel cuya clave consistirá en un código de provincia, entonces, los nodos hijos serán los municipios cuya clave foránea de provincia coincida con el valor de esa provincia.

Ejemplo:

applicationContext-services.xml

```
<bean id="abecedarioNodesService"
class="atlas.samples.service.tree.impl.AbecedarioNodesService" />

<bean id="provinciaLetraNodesService"
class="atlas.samples.service.tree.impl.ProvinciaLetraNodesService">
      <property name="provinciaSucaDAO">
                  <ref bean="provinciaSucaDAO" />
            </property>
</bean>

<bean id="municipioNodesService"
      class="atlas.samples.service.tree.impl.MunicipioNodesService">
      <property name="provinciaSucaDAO">
            <ref bean="provinciaSucaDAO" />
      </property>
</bean>
```

- Las referencias: ***abcedarioNodesService***, ***provinciaLetraNodesService*** y ***municipioNodeService***. son los servicios creados para cada nivel. Deben implementar la interfaz (***atlas.componentes.service.tree.NodesService***)

- Estos objetos deben de proveerse de clases de acceso a datos (DAO) para obtener sus datos. De esta manera se mantiene una arquitectura de tres capas. Estas clases tienen que tener los métodos necesarios para que los servicios ***abcedarioNodesService***, ***provinciaLetraNodesService*** y ***municipioNodeService*** devuelvan en sus métodos de la interfaz ***NodesService*** los elementos correspondientes a cada nivel.

- La lista devuelta por los servicios debe contener elementos que implementen la interfaz ***atlas.componentes.service.tree.Node***. A continuación se describe dicha interfaz.

3.2.2. Nodos

Los servicios deben devolver una lista de elementos que cumplan unos requisitos en forma de interfaz. Hay dos interfaces definidas, debiendo implementar una u otra en función del modo de uso del árbol.

- ***atlas.componentes.service.tree.Node***: Es la interfaz básica de un nodo e incluye los métodos:
    - String getNombre(); debe devolver la etiqueta del nodo que se mostrará en el árbol.
    - Object getClave(); debe devolver el identificador del nodo, que utilizará el servicio para obtener los hijos una vez el usuario lo expanda.

  Estos métodos son suficientes para que la clase controladora pueda gestionar el árbol, pero mediante una propiedad es posible indicarle al componente que muestre después de cada nodo el número de hijos, en caso de usar esta funcionalidad la interfaz que el nodo debe implementar es:
- ***atlas.componentes.service.tree.NodeCount***: Esta interfaz hereda de *Node*, y por tanto también se debe cumplir, pero además incluye los siguientes métodos.
    - Integer getNumeroHijos();
    - void setNumeroHijos(Integer numeroHijos);

  Para que el árbol funcione correctamente ambos métodos deben actualizar una variable de tipo Integer.

La librería de componente incluye implementación básica de *NodeCount*, la clase ***atlas.componentes.service.tree.impl.NodeImpl***.

Más adelante veremos cómo el componente permite ampliar la información que se muestra, de modo que se pueda mostrar información diferente al nombre del nodo.

4. USO

Una vez instalado el módulo de componentes puede procederse a su utilización. Para ello deben realizarse los pasos indicados en los siguientes apartados:

4.1. Paso 1: Definición del espacio de nombres de etiquetas de Atlas

Es necesario crear un fichero *xhtml* y establecer la definición del espacio de nombres para las etiquetas de componentes de Atlas. Un ejemplo de cabecera de fichero *xhtml* es la siguiente:

Fichero.xhtml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es"
      xmlns:ui="http://java.sun.com/jsf/facelets"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core"
      xmlns:atlas="http://atlas.core.componentes/jsf">
```

4.2. Paso 2: Inserción de la etiqueta en la página

Para invocar al componente hay que crear una página XHTML donde se inserte la etiqueta de este componente.

Es importante que el componente esté incluido dentro de un formulario, si no lo está no funcionará correctamente.

En la propiedad *dataModel* del componente es necesario establecer una EL (Expression Language) que indica el método del backing bean que devuelve el bean de tipo `atlas.componentes.bean.AtlasTree`.

Fichero.xhtml

```
<atlas:arbol id="arbol" dataModel="#{provinciasFormBean.atlasArbol}"
    services="abecedarioNodesServices,provinciaLetraNodesService,
    municipioNodesService"/>
```

Un ejemplo para este método (de la clase de backing bean ProvinciasFormBean) podría ser:

ProvinciasFormBean.java

```
public class ProvinciasFormBean extends BaseBean {

      . . .

    private AtlasTree arbol = new AtlasTree();

    public AtlasTree getArbol() {
        Return arbol;
    }

    public void setArbol(AtlasTree arbol) {
        this.arbol = arbol;
    }

      . . .

}
```

Los atributos del componente son los siguientes:

Nombre atributo	Obligatorio	Descripción
Id	SI	Identificador del componente. Debe ser único
dataModel	SI	EL (Expression Language) al bean que controla el funcionamiento del árbol. El objeto devuelto debe ser de tipo "atlas.componentes.bean.AtlasTree"
services	SI	Lista con los identificadores en el contexto de Spring del servicio o servicios encargados de generar los diferentes niveles del árbol.
lazyMode	NO	Valor true/false para indicar si el componente debe consultar los hijos de los nodos una vez abierto un nivel. Si es false se hará una búsqueda en los nodos recién abiertos para ver cuáles tienen hijos y cuáles son nodos hoja. Si su valor es true la propiedad mostrarNumeroNodosHijos no tiene efecto. Su valor por defecto es false.
cacheMode	NO	Valor true/false para indicar
mostrarNumeroNodosHijos	NO	Valor true/false si se quiere mostrar el número de hijos de cada nodo. Su valor por defecto es true.
claveInicial	NO	Clave inicial para el nodo raíz del árbol.
iconoCerrado	NO	url de la imagen para el icono de nodo cerrado.
iconoAbierto	NO	url de la imagen para el icono de nodo abierto.
IconoHoja	NO	url de la imagen para un nodo hoja.
rendered	NO	Valor true|false para seleccionar si este componente se va a mostrar. Por defecto es true.
styleClass	NO	Estilo para el componente.

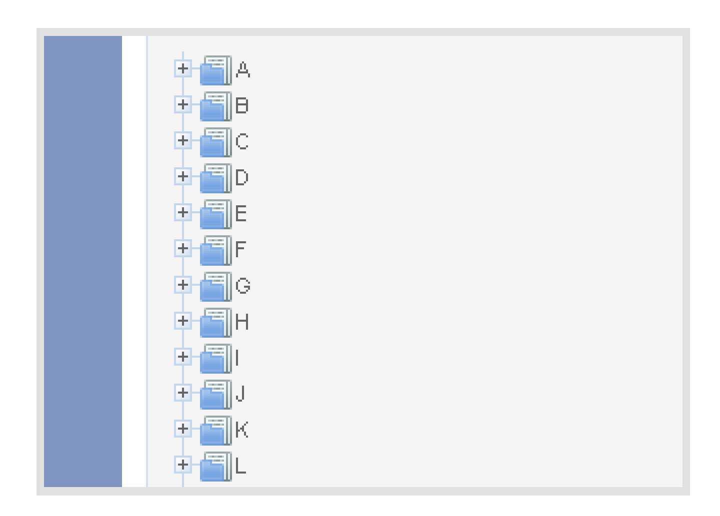
4.3. Carga de nodos y estado del árbol

En este apartado se explica un poco más el modo en que se obtienes los nodos de los diferentes niveles, el número de llamadas a los servicios y qué información guarda internamente el componente. Para este caso dos propiedades de la tabla anterior son las encargadas de definir el modo en que se comporta el componente: *lazyMode* y *cacheMode*.

Como se indica en la descripción de ambas, sus valores por defecto son *false* y *true* respectivamente, de modo que

el árbol inicialmente obtendrá los hijos de un nodo y hará una llamada al servicio del siguiente nivel para cada nodo hijo, mostrando estos nodos como nodos hoja o no según corresponda, y, en caso en que así se configure, mostrar también el número de hijos de cada nodo. Además, cuando se contraiga un nodo y vuelva a expandirse no se volverá a llamar al servicio para obtener los hijos sino que se mostrarán los hijos ya obtenidos previamente, y en el mismo estado en que estuviera el sub-árbol cuando se cerró el nodo.

Estos comportamientos se pueden modificar con ambas propiedades. Si se desea minimizar el número de consultas se puede establecer la propiedad *lazyMode* a true, en este caso sólo se hará una consulta al servicio del nodo expandido para obtener el listado de hijos, sin consultar si los hijos son nodo hoja o no, visualmente se traducirá en que todos los nodos del nivel abierto se mostrarán como nodos con hijos hasta que se expanda cada uno de ellos, momento en el que se abrirá el nivel o se cambiará el icono a nodo hoja si no hubiera hijos que mostrar. Esto ocurrirá salvo en el caso en que se especifique un número determinado de niveles, en este caso cuando se llegue al último nivel se mostrarán todos los nodos como nodos hoja.



Por otro lado en casos en que la estructura del árbol pueda cambiar, es necesario que siempre que se expanda un nodo, se obtengan del servicio sus hijos, independientemente de si este nodo ya ha sido visitado o no, esto se puede hacer especificando la propiedad *cacheMode* a *false*. Para este caso hay que aclarar que cada vez que se expanda un nodo, la estructura será con todos los nodos hijos cerrados, es decir, se pierde la estructura que tuviera cuando se cerró el nodo.

4.4. Interacción con el árbol

Relacionado con el apartado anterior, es posible interactuar con el árbol desde otros componentes de la página o desde un backing bean. Para esto, se ha definido un API de operación en la clase AtlasTree, estos métodos se deben ejecutar en el backing bean al ejecutar una acción. Los métodos son los siguientes:

- *recargar()*: recarga las información del árbol, volviendo al estado inicial.
- *recargarNodo(List<Object> listaClaves)*: recarga la información de un nodo. Para especificar el nodo se debe pasar como parámetro una lista con las claves hasta llegar al nodo.

- *expandirNodo(List<Object> listaClaves)*: expande un nodo. Los nodos de niveles superiores deben estar ya abierto, de otro modo no tiene efecto.
- *expandirNodoRuta(List<Object> listaClaves)*: expande un nodo y los niveles previos si no han sido abiertos.
- *cambiarClaveInicial(Object clave)*: modifica la clave inicial del árbol.
- *cerrarNodos()*: cierra todos los nodos de primer nivel abiertos. Si la propiedad *cacheMode* es false el resultado es igual a la función *cerrarNodosRecursivo*.
- *cerrarNodosRecursivo()*: cierra todos los nodos abiertos en el árbol.
- *cerrarNodo(List<Object> listaClaves)*: Cierra un nodo.
- *cerrarNodoRecursivo(List<Object> listaClaves)*: Cierra todos los nodos abiertos a partir de uno.
- *expandirNodoInicial(List<Object> listaClaves)*: Se añade el nodo a una lista de los que deben mostrarse al inicializar el árbol (en ese momento es cuando se consulta la lista). Esta función puede ser llamada varias veces y está pensada para ejecutarse al cargar el componente por primera vez, de modo que se pueda establecer un estado inicial en el que algunos nodos aparezcan abiertos al cargar la página. El comportamiento a la hora de abrir los nodos es el de la función *expandirNodoRuta*, de modo que abre el nodo indicado y los niveles previos hasta él.

Si el componente que ejecuta la acción es ajax hay que especificar el id del componente árbol en la propiedad *render* para que se actualice correctamente.

4.5. Personalización de los nodos

El componente permite personalizar los nodos del árbol. Por defecto se muestra un texto con el nombre del nodo, pero mediante el tag de facelets *define* con el nombre *nodo* se puede sustituir este texto por cualquier otro o incluir cualquier otro componente.

En el siguiente ejemplo se sustituye el nodo por un commandLink con el nombre como texto.

Fichero.xhtml

```
<atlas:arbol id="arbol" dataModel="#{provinciasFormBean.atlasArbol}"
  services="abecedarioNodesServices,provinciaLetraNodesService,
  municipioNodesService">
  <ui:define name="nodo">
    <a4j:commandLink actionListener="#{provinciasFormBean.selectNode}"
      value="#{item.data.nombre}" execute="@this" render="panelNodo">
        <f:param name="nodoSeleccionado" value="#{item.data.nombre}" />
    </a4j:commandLink>
  </ui:define>
<atlas:árbol/>
```

Para acceder a la información básica de un nodo: clave, nombre y número de hijos (en caso de usar esta funcionalidad), se deben usar con los siguientes identificadores: *item.data.clave*, *item.data.nombre*,

item.data.numeroHijos respectivamente.

También se puede hacer referencia a cualquier otra propiedad de la clase que representa el nodo, aunque no forme parte de las interfaces *Node* y *NodeCount*. Para acceder a la clase se debe usar al igual que en el caso anterior *ítem.data*, esto da acceso al nodo, a partir de ahí se puede acceder a cualquier propiedad.

4.6. RECOMENDACIONES Y BUENAS PRÁCTICAS

Para el buen funcionamiento del componente hay que seguir los pasos de instalación, configuración de ficheros y implementación de clases tal y como se ha indicado en los anteriores puntos.

4.7. EJEMPLO DE USO

Se puede ver un ejemplo de dicho componente navegando en la aplicación componentes de la siguiente manera:

Inicio > Otros componentes Atlas > Árbol

5. PREGUNTAS MÁS FRECUENTES

En este apartado se debe incluir una lista de preguntas más frecuentes sobre el objetivo del documento, así como las respuestas pertinentes.

Pregunta: ¿Dónde puedo encontrar información general sobre los componentes?
Respuesta: En la aplicación de demostración de los componentes del Framework Atlas

Pregunta: ¿Cómo se ha implementado el componente en su capa de presentación?
Respuesta: Mediante un componente de Facelets. Hace uso además de un componente de árbol de RichFaces.

Pregunta: ¿Cómo puedo modificar los estilos del componente?
Respuesta: Mediante los atributos de estilo del componente. Para más información consultar la tabla de atributos en la sección de *Uso* del componente.

6. ENLACES RELACIONADOS

Producto	URL
Ajax4JSF	http://www.jboss.org/jbossrichfaces/
Barbecue	http://barbecue.sourceforge.net/
Commons BeanUtils	commons.apache.org/beanutils/
Commons Configurations	http://commons.apache.org/configuration/
Facelets	https://facelets.dev.java.net/
Hibernate	http://www.hibernate.org/
Hibernate Annotations	http://www.hibernate.org/hib_docs/annotations/reference/en/html_single/
JAXB	http://java.sun.com/webservices/jaxb/
Jcaptcha	jcaptcha.sourceforge.net/
JPA	http://java.sun.com/developer/technicalArticles/J2EE/jpa/
JSF	http://java.sun.com/javaee/javaserverfaces/
JSFUnit	http://www.jboss.org/jsfunit/
Log4J	http://logging.apache.org/log4j/
MyFaces Core	http://myfaces.apache.org/
RichFaces	http://www.jboss.org/jbossrichfaces/
Spring	http://www.springframework.org/
Spring Security	http://www.springframework.org/
Tomahawk	http://myfaces.apache.org/tomahawk/
Velocity	http://velocity.apache.org/